Ing.Dušan STREIT
Slezan n.p. Frýdek-Místek

PARAMETRICKÉ PROGRAMY UŽIVATELSKÉHO SOFTWARE V SYSTÉMU ZPRACOVÁNÍ DAT

1. Úvod

Havířovský seminář se původně nazýval "Metody programování počítačů 3. generace". Protože organizátoři semináře předvídavě předpokládali, že přijde den, kdy budou překonány nebo naplněny vytýčené problémové oblasti
- programovací techniky, metody a prostředky,
- tvorba a údržba programové dokumentace a standardizace,
- styk programátora s okolím a organizace práce,
- výuka a výchova programátorů

formulované v roce 1975 v souvislosti s nástupem počítačů 3. generace, došlo počínaje rokem 1978 ke změně názvu semináře. O rok později už bylo zřejmé, že takový den nadchází.

Je jistě zásluhou semináře, že se dnes nebudeme přít o používání GO TO ve strukturovaném programování, o prospěšnosti normovaného programování, modularity, standardizace a týmové práce. Jelikož mnohé myšlenky, přednesené v roce 1975 a zdající se příliš odvážné, jsou v roce 1980 samozřejmostí, otázka zní jinak: Co dál?

Mnohými ústy byla na semináři v roce 1979 tato otázka vyslovena, jako nemělé pozeptání i jako varovná výzva; nazval bych ji nosnou otázkou semináře "Programování 79". V diskuzi se na ni v několika tématických okruzích snažili účastníci nalézt odpověď.

Kdybych měl na tuto otázku obrazně odpovědět, předpokládám přesun diskutujících z loni nejpočetnějšího tématického okruhu věnujícího se technologii programování do okruhu zabývajícího se typovými prvky a parametrickými programy. Vycházím z vlastní zkušenosti - i já jsem prošel tímto vývojem. Promiňte, že jsem se sám zvolil jako reprezentativní prvek, ale můj odborný růst v oblasti software se časově kryje s vývojem semináře.

Důkladná aktivní znalost metod a technik programování je totiž základním předpokladem typizace a parametrizace. Parametrický program jako nejlepší typový prvek je na druhé straně zákonitým důsledkem důsledného uplatnění moderních programovacích metod a technik (např. generátor normovaného programování). Jejich uplatněním dochází k potlačování subjektivních přístupů, k typizaci a k možnosti zobecnění a modifikovatelnosti parametry.

Ještě v úvodu bych se chtěl omluvit těm, kteří hledají řešení otázky "Co dál?" v databankách a interaktivním zpracování. S hlubokou úctou se před nimi skláním a doufám, že snad také někdy - až vývoj mé programátorské osobnosti dále pokročí - se přes ty parametrické programy dopracuji až mezi ně. Myslím, že pro interaktivní zpracování jsou parametrické programy tou nejlepší průpravou, i když "jen" v dávkovém režimu. Zatím tuto záležitost nechávám na firemních a ústavních kapacitách.

Dále bych se chtěl omluvit za poněkud teoretické ladění mého příspěvku. Sleduji však jeden praktický cíl: "Dokázat, že parametrické programy uživatelského software

jsou na rutinní úrovni schopny programově zabezpečit podstatnou část systému zpracování dat." Těm, kteří parametrické programy degradují na prostředek pro jednorázové aplikace, se už omluvit nechci.

2. Co jsou to parametrické programy?

Parametrické programy jako odraz zákonitého vývoje jsou jen dalším stupněm při prosazování 2 tendencí v programování:

1) zvyšování úrovně abstrakce a syntaxe,
2) vytváření specializovaných programových prostředků.

Prvá tendence je zřejmá ze vzniku vyšších programovacích jazyků. Mezi programátora a hardware počítače byl vložen prostředník - kompilátor. I když kompilátor není parametrický program ve smyslu tohoto příspěvku, na určité úrovni abstrakce jsou zdrojové programy vlastně parametry pro kompilátor. Této úrovni abstrakce odpovídá i příslušný stupeň specializace. Vznikly programovací jazyky pro hromadné zpracování dat, pro vědeckotechnické výpočty, pro simulační systémy atd.

Dalším prostředníkem, který nastupuje později, je operační systém - také v podstatě parametrický modulární program. A parametrizace pokračuje dále. Dalším prvkem, který odděluje programátora a hardware je generátor, případně interpretátor. Také této úrovni abstrakce a syntaxe odpovídá příslušný stupeň specializace: vznikají parametrické systémy pro konverze, třídění, výběry, tisky atd.

Základním předpokladem těchto systémů je účelovost; právě tato účelovost umožňuje zúžit problém natolik, aby bylo možno vytvořit specializovaný, avšak obecný a parametrizovatelný programový prostředek pro jeho řešení.

Předpokladem určité úrovně zobecnění problému je na druhé straně standardizace. I na firemním software vidíme snahu standardizovat např. vstupní a výstupní operace (I/O Generátor, IOCS, MTH systém, UDAS atd.). V podmínkách jednotlivých výpočetních středisek však můžeme jít ve standardizaci ještě dále. Můžeme standardizovat např. formáty u děrné pásky, délku bloku u magnetické pásky, formáty tiskových sestav atd.

Parametrický program není sémanticky prázdný, jak se na určité úrovni abstrakce jeví např. kompilátor. Obsahuje typové řešení určitého problému. Navenek musí být funkční, parametrický "jazyk" by měl být neprocedurální.

Parametrický program je nejlepší typový prvek, k čemuž jej předurčuje další základní předpoklad - a to flexibilita. Jedná se o flexibilitu vnější (obměnou parametrů a okolí - tzv. variabilita a portabilita) i vnitřní (možnost rozšíření funkcí - tzv. adaptibilita). Vnitřní flexibility nejlépe dosáhneme modularitou parametrického programu.

Parametrické řešení problému lze realizovat vícestupňově: Parametricky lze vytvářet objekt - jiný parametrický program (např. generování operačního systému) nebo subjekt - parametry pro jiný parametrický program. Rozvedeno až k absurdní krajnosti můžeme si představit tuto řadu: generátor operačního systému - operační systém - generátor programů - preprocessor - kompilátor - interpretátor/simulátor/aplikační program - parametrický modul.

Parametrické vytváření objektu se používá tam, kde se jedná o přizpůsobení objektu (parametrického programu) konkrétním podmínkám. V zásadě se jedná o výběr modulů. Parametrické vytváření subjektu (parametrů) se využívá při snaze přiblížit některé funkce uživateli - laikovi. Pro tvorbu parametrů vytvoříme mezistupeň, který může

pracovat např. na základě makroinstrukcí.

Každý stupeň parametrizace směrem od počítače nám sice vymezuje užší prostor, na druhé straně nám však umožňuje využít zpředmětnělých znalostí specialistů. Jistě dnes netouží nikdo po tom, programovat pro každou aplikaci třídicí programy. Stejně tak by bylo neefektivní programovat přímo v binárním kódu, nehledě na to, že některé instrukce jsou stejně extrakódy a slouží vlastně jako "parametry" Supervisoru a většina funkcí (u některých procesorů dokonce velmi syntetických) je trvale "naprogramována" v paměti ROM.

3. Základní typy parametrických programů

Nebudu zde rozvádět dobře známé základní rozdělení parametrických programů na interpretátory a generátory [1]. Chtěl bych uvést pouze několik poznámek.

Interpretátor v průběhu zpracování interpretuje parametry. Nemusí to však vždy v praxi znamenat neustálé prohledávání tabulek. Kontrola a také interpretace parametrů může být částečně oddělena a tvořit samostatný segment interpretátoru. V tomto segmentu je možno provádět modifikaci programu na základě tvorby indexních slov a různých substitucí. Může to vést až k tomu, že interpretátor obsahuje segment pro generování binárního kódu (tzv. translator).

Generátor naproti tomu vytváří konkrétní aplikační programy ve zdrojové formě. Ale lze generovat taky parametry nebo přímo "object" programy (zde se vlastně jedná už o kompilaci). Existují také různé úrovně generátorů - od preprocessorů vlastně až po kompilátory.

Preprocessor provádí modifikaci konkrétního programu

ve zdrojové formě na základě speciálních výroků. Může jím být také realizován hostitelský jazyk např. pro využití rozhodovacích tabulek. K těmto obecně pojatým generátorům patří i takové, které generují pouze zdlouhavé a opakující se sekvence výroků nebo příkazů např. COBOLu a realizují je na způsob makroinstrukcí [7], což může vést až ke vzniku makrogenerátoru jako součásti kompilátoru. Určitým způsobem do této kategorie patří i generátor normovaného programování [4], [5], [6].

Chtěl bych však obrátit pozornost zvláště na ty generátory, které obsahují typizované programové moduly reprezentující obsahovou kostru standardizovaného problému [2]. Oproti předchozí skupině generátorů je logika generovaného programu vytvářena z větší části generujícím programem a jen v omezené míře parametry. Velmi výhodné je i při této koncepci zachovat možnost vkládat uživatelské procedury přímo v programovacím jazyku generovaného programu, zvláště pokud se jedná o vyšší programovací jazyk.

Výhody generátorů oproti interpretátorům jsou zřejmé hlavně v oblasti, o kterou mi jde- při rutinním programovém zabezpečení systému zpracování dat z hlediska uživatele - programátora. Při opakovaném zpracování se vyplatí ztratit strojový čas při jednorázovém generování. (Jedná se o minimální ztráty, zvláště při dávkovém generování.) Domnívám se, že rovněž programování generátorů je jednodušší než programování interpretátorů. Významným způsobem nám totiž při realizaci napomáhá kompilátor.

Interpretační programy v této oblasti mohou být výhodné při zabezpečení jednoduchých nebo některých speciálních aplikací např. pro konverze, různé převody, jednoduché kontroly, aktualizace nebo pro některé matematické aplikace. Jistě nebudeme vytvářet programovací jazyky a realizovat je prostřednictvím interpretátorů (abstrahuji

se od interaktivního zpracování).

Naproti tomu program vygenerovaný podle všech zásad, které platí při běžném programování, může směle soutěžit s konkrétním aplikačním programem. Ve většině případů jej i předčí, nehledě na to, že čas využití centrální jednotky při současných rychlostech není zdaleka rozhodující. Naopak takový program bývá vygenerován na principech strukturovaného a modulárního programování a jako takový mívá i "výchovný" dopad na programátory.

4. Některé praktické zkušenosti

V našem VS jsme se rozhodli celý skelet systému zpracování dat programově zabezpečit parametrickými programy uživatelského software [2]. Za velmi důležitý předpoklad aplikace parametrických programů v této oblasti považuji uplatnění systémového přístupu a postupu shora dolů. V praxi to znamená nevyrábět parametrické programy živelně a jednotlivě, ale systém zpracování dat dekomponovat na programové chody, a ty ve vzájemných souvislostech v maximální možné míře a s přihlédnutím ke kriteriu optimality zabezpečit parametrickými programy.

Dále bych chtěl uvést některé praktické příklady aplikace jednotlivých parametrických programů:

Programovali jsme např. program pro prvotní kontroly vstupních dat [3]. Program samotný byl naprogramován v jazyku COBOL, jednotlivé kontrolní moduly pro jednotlivé typy kontrol byly naprogramovány v jazyku PLAN. Samotný program vlastně parametrický nebyl, jednalo se o výběr kontrolních modulů a naplnění formálních parametrů skutečnými parametry při vyvolávání kontrolních modulů. Ve zdrojové formě je výhodné tento problém řešit na úrovni preprocessoru.

Je však možno jej řešit také na binární úrovni zpracováním modulu, který čte parametry a předává je příslušným modulům. Program funguje tedy jako interpretátor.

Výhodou prvého způsobu je vyšší flexibilita a nižší nároky na paměť (program obsahuje jen skutečně potřebné moduly podle potřeby i jednoúčelové), druhý způsob je zase pohotovější a s výhodou se využívá u jednorázových aplikací.

Další aplikací parametrických programů, kterou bych chtěl uvést jako příklad, bylo vytvoření generátoru pro sdružování a výběr dat (tzv. selektivní syntéza) [2]. Jelikož byl tento problém standardizován, bylo možno typizovat i jeho řešení. Výsledek této typizace představuje skelet zdrojového programu selektivní syntézy. Tento skelet je možno modifikovat různými popisy souborů, podmínkami výběru, případně dalšími procedurami uživatele (které vycházejí z bloků normovaného programování).

Skelet i parametry jsou založeny na bázi jazyka COBOL. Skelet je pak apriorně nahrán jako součást generátoru; generátor provádí ze skeletu výběr těch operací, které se vztahují k definovaným souborům. Syntézou skeletu a parametrů se generuje modifikovaný program.

Z parametrů jsou doplňovány nebo generovány chybějící části skeletu do modifikovaného programu. Je to umožněno systémem referenčních čísel, kterými je skelet rozčleněn. Referenční čísla označují místa ve skeletu, kam jsou přenášeny korespondující výroky a příkazy z parametrů, které jsou uvozeny shodným referenčním číslem. Takto je možno do programu vkládat celé uživatelské procedury.

Tímto systémem generování lze výsledný program modifikovat nejen na úrovni parametrů, ale i na úrovni skeletu (tzv. vnitřní flexibilita). Tak např. došlo na základě

zkušeností k doplnění možnosti tisků a kumulací přímo do skeletu.

Další výhodou např. vůči generátoru normovaného programování je to, že skelet selektivní syntézy je oproti obecnému vývojovému diagramu normovaného programu "sémanticky plnější". Znamená to, že není normována pouze forma, ale je typizován i obsah (bez nutnosti zadávat makrodefinice.

5. Závěr

Obdobnými aplikacemi parametrických programů se nám podařilo zabezpečit uživatelským software systém zpracování dat asi ze 40 - 45 %. Předpokládáme, že uvedené procento se bude dále zvyšovat. Přitom jsme nezaznamenali žádný pokles efektivity programů, naopak jsme dosáhli těchto přínosů:

1) Zvýšení produktivity práce programátora
2) Snížení počtu chyb pramenících z běžného programování (tím i úspora strojového času počítače na jejich odstranění při kompilacích a laděních)
3) Možnost využít při aplikaci méně kvalifikované programátory - kódovače
4) Odpovídající úspora práce a materiálu při děrování programů
5) Zrychlení cyklu:
   analýza a požadavky - projektování a programování - - automatizované zpracování na počítači a obdržení výsledků
6) Vytvoření podmínek pro uplatnění rozhodovacích tabulek
7) Vytvoření pseudo - báze dat pro monitorový systém v

režimu vícenásobného přístupu a interaktivního zpracování

8) Hlubší uplatnění funkčního hlediska při tvorbě technických projektů a zlepšení komunikace mezi analytiky a programátory

9) Parametrické zabezpečení systému zpracování dat navíc přerůstá rámec programování a předpokládá a stimuluje racionalizaci technologie zpracování dat a rozšíření standardizace

10) V konečném důsledku se body 1 až 9 promítnou ve zvýšení celkové efektivnosti ASŘP

Byl bych rád, kdyby celkový závěr mého příspěvku vyplýval z jeho obsahu a odpovídal již v úvodu prezentovanému záměru. Jestliže se mi nepodařilo přesvědčit ty, kteří parametrické programy nadále považují pouze za pomocníka při jednorázových aplikacích, budu považovat tento příspěvek jako prospěšný tehdy, když vzbudím zájem o parametrické programy u těch, kteří dosud neměli vyhraněný názor, tak jako já jsem byl podnícen některými příspěvky z roku 1975.

Mimo uvedené literatury a zkušeností z n. p. SLEZAN ve Frýdku-Místku čerpal tento příspěvek i z diskuze skupiny zabývající se typovými prvky a parametrickými programy na semináři "Programování 79" v Havířově.

L i t e r a t u r a :

[1] Jiříček Petr: Generátory, přednosti a nevýhody této techniky, sborník Metody programování počítačů 3. generace, DT ČVTS Ostrava, 1975

[2] Streit Dušan: Selektivní syntéza setříděných sériových souborů ..., sborník Programování 79, DT ČsVTS Ostrava, 1978

[3] Streit Dušan: Zkušenosti se standardizací v oblasti konverzí a prvotních kontrol vstupních dat, sborník Metody programování počítačů 3. generace, DT ČVTS Ostrava, 1977

[4] Metzl Karel: Generátor zdrojových programů, sborník Metody programování počítačů 3. generace, DT ČVTS Ostrava, 1976

[5] Baláž Stan.: Generátor normového programování a NPG, sborník Metody programování počítačů 3. generace. DT ČVTS Ostrava, 1976

[6] Drobný: Generátor zdrojových cobolských programů SGNP, SHR Most, 1977

[7] Volák Jiří: Generátor cobolských programů CMCGEN, sborník Programování 79, DT ČVTS, 1979